BDA_Lab2

Olayemi Morrison (olamo208) and Greeshma Jeev Koothuparambil(greko370)

2024-05-16

For each exercise, include the following data in the report and sort it as shown:

Question 1

year, station with the max, maxValue ORDER BY maxValue DESC
year, station with the min, minValue ORDER BY minValue DESC

```
from pyspark import SparkContext
from pyspark.sql import SQLContext, Row, SparkSession
from pyspark.sql import functions as F

sc = SparkContext(appName="BDALab2")
sqlContext = SQLContext(sc)

temp_reading = sc.textFile("BDA/input/temperature-readings.csv")
lines = temp_reading.map(lambda line: line.split(";"))

# (key, value) = (year,temperature)
yearTemp_reading = lines.map(lambda p: Row(station=p[0],
date=p[1], year=p[1].split("-")[0],time=p[2],value=float(p[3]),quality=p[4]))

schemayearTemp_reading = sqlContext.createDataFrame(yearTemp_reading)
schemayearTemp_reading.registerTempTable("yearTemp_reading")

minYearTempSta =
schemayearTemp_reading.where((schemayearTemp_reading["year"] >= 1950) &
(schemayearTemp_reading["year"] <= 2014)). \
groupBy('year', 'station'). \
agg(F.min('value').alias('annualMin')). \
select('year', 'station', 'annualMin')
# orderBy(['annualMin'], ascending=[False])

TempMinByYear = schemayearTemp_reading.groupBy('year'). \
agg(F.min('value').alias('annualMin'))

TempMinByYearWithStation =
minYearTempSta.join(TempMinByYear, on=['year', 'annualMin'])\
.select('year', 'station', 'annualMin')\
.orderBy(['annualMin'], ascending=[False])

# max
maxYearTempSta =
schemayearTemp_reading.where((schemayearTemp_reading["year"] >= 1950) &
(schemayearTemp_reading["year"] <= 2014)). \
groupBy('year', 'station'). \
agg(F.max('value').alias('annualMax')). \
select('year', 'station', 'annualMax')


TempMaxByYear = schemayearTemp_reading.groupBy('year'). \
agg(F.max('value').alias('annualMax'))

TempMaxByYearWithStation =
maxYearTempSta.join(TempMaxByYear, on=['year', 'annualMax'])\
.select('year', 'station', 'annualMax')\
.orderBy(['annualMax'], ascending=[False])

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
TempMaxByYearWithStation.rdd.saveAsTextFile("BDA/output/max")
TempMinByYearWithStation.rdd.saveAsTextFile("BDA/output/min")
```

Result

max:
Row(year='1975', station='86200', annualMax=36.1)
Row(year='1992', station='63600', annualMax=35.4)
Row(year='1994', station='117160', annualMax=34.7)
Row(year='2014', station='96560', annualMax=34.4)
Row(year='2010', station='75250', annualMax=34.4)
Row(year='1989', station='63050', annualMax=33.9)
Row(year='1982', station='94050', annualMax=33.8)
Row(year='1968', station='137100', annualMax=33.7)
…

min:
Row(year='1990', station='166870', annualMin=-35.0)
Row(year='1990', station='147270', annualMin=-35.0)
Row(year='1952', station='192830', annualMin=-35.5)
Row(year='1974', station='179950', annualMin=-35.6)
Row(year='1974', station='166870', annualMin=-35.6)
Row(year='1954', station='113410', annualMin=-36.0)
Row(year='1992', station='179960', annualMin=-36.1)
…

Question 2

year, month, value ORDER BY value DESC year, month, value ORDER BY value DESC

```
temp_reading = sc.textFile("BDA/input/temperature-readings.csv")
lines = temp_reading.map(lambda line: line.split(";"))

yearTemp_reading = lines.map(lambda p: Row(station=p[0],
                                           date=p[1],
                                           year=p[1].split("-")[0],
                                           month=p[1].split("-")[1],
                                           time=p[2],
                                           value=float(p[3]),
                                           quality=p[4]))
```

```
schemayearTemp_reading = sqlContext.createDataFrame(yearTemp_reading)



# 2.1
temperature_count = schemayearTemp_reading.where(
    (schemayearTemp_reading["year"] >= 1950) & (schemayearTemp_reading["year"] <= 2014) &
    (schemayearTemp_reading["value"] >= 10)) \
    .select('year', 'month') \
    .groupBy('year', 'month') \
    .count() \
    .orderBy('count', ascending=[False])

# 2.2
temperature_countFromStation = schemayearTemp_reading.where(
    (schemayearTemp_reading["year"] >= 1950) & (schemayearTemp_reading["year"] <= 2014) &
    (schemayearTemp_reading["value"] >= 10)) \
    .select('year', 'month', 'station') \
    .groupBy('year', 'month', 'station') \
    .count() \
    .groupBy('year', 'month').count() \
    .orderBy('count', ascending=[False])

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
temperature_count.rdd.saveAsTextFile("BDA/output/1")
temperature_countFromStation.rdd.saveAsTextFile("BDA/output/2")
```

Result

Answer 2.1:
Row(year=u'2014', month=u'07', count=147910)
Row(year=u'2011', month=u'07', count=147060)
Row(year=u'2010', month=u'07', count=143860)
Row(year=u'2012', month=u'07', count=138166)
...

Answer 2.2:
Row(year=u'1972', month=u'10', count=378)
Row(year=u'1973', month=u'06', count=377)
Row(year=u'1973', month=u'05', count=377)
Row(year=u'1972', month=u'05', count=376)
Row(year=u'1973', month=u'09', count=376)
Row(year=u'1972', month=u'08', count=376)
...

Question 3

year, month, station, avgMonthlyTemperature ORDER BY avgMonthlyTemperature DESC

```
temp_reading = sc.textFile("BDA/input/temperature-readings.csv")
lines = temp_reading.map(lambda line: line.split(";"))

yearTemp_reading = lines.map(lambda p: Row(station=p[0],
                                           date=p[1],
                                       year=p[1].split("-")[0],
                                       month=p[1].split("-")[1],
                                       time=p[2],
                                       value=float(p[3]),
                                       quality=p[4]))

schemayearTemp_reading = sqlContext.createDataFrame(yearTemp_reading)

DailyTemperature = schemayearTemp_reading.where((schemayearTemp_reading["year"] >= 1950) &
(schemayearTemp_reading["year"] <= 2014)) \
    .groupBy('year', 'month', 'date', 'station') \
    .agg(F.min('value').alias('min'), F.max('value').alias('max'))

avgDailyTemperature = DailyTemperature.withColumn('avgDailyTemperature',
                    (DailyTemperature['min'] + DailyTemperature['max']) * 0.5)

avgMonthlyTemperature =
avgDailyTemperature.select('year', 'month', 'station', 'avgDailyTemperature') \
    .groupBy('year', 'month', 'station') \
    .agg(F.avg('avgDailyTemperature').alias('avgMonthlyTemperature')) \
    .orderBy('avgMonthlyTemperature', ascending=[False])

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
avgMonthlyTemperature.rdd.saveAsTextFile("BDA/output/A2Q3")
```

Result

```
Row(year=u'2014', month=u'07', station=u'96000', avgMonthlyTemperature=26.3)
Row(year=u'1994', month=u'07', station=u'96550', avgMonthlyTemperature=23.071052631578947)
Row(year=u'1983', month=u'08', station=u'54550', avgMonthlyTemperature=23.0)
Row(year=u'1994', month=u'07', station=u'78140', avgMonthlyTemperature=22.970967741935482)
Row(year=u'1994', month=u'07', station=u'85280', avgMonthlyTemperature=22.87258064516129)
Row(year=u'1994', month=u'07', station=u'75120', avgMonthlyTemperature=22.858064516129033)
Row(year=u'1994', month=u'07', station=u'65450', avgMonthlyTemperature=22.856451612903232)
```

Question 4

station, maxTemp, maxDailyPrecipitation ORDER BY station DESC

```
temp_reading = sc.textFile("BDA/input/temperature-readings.csv")
precipitation_file = sc.textFile("BDA/input/precipitation-readings.csv")

lines = temp_reading.map(lambda line: line.split(";"))

yearTemp_reading = lines.map(lambda p: Row(station=p[0],
                                       date=p[1],
                                       year=p[1].split("-")[0],
                                       month=p[1].split("-")[1],
                                       time=p[2],
                                       value=float(p[3]),
                                       quality=p[4]))

schemayearTemp_reading = sqlContext.createDataFrame(yearTemp_reading)
```

```
lines = precipitation_file.map(lambda line: line.split(";"))

precipReadings = lines.map(lambda p: Row(station=p[0],
                                        date=p[1],
                                        year=p[1].split("-")[0],
                                        month=p[1].split("-")[1],
                                        time=p[2],
                                        value=float(p[3]),
                                        quality=p[4]))

schemaPrecipReadings = sqlContext.createDataFrame(precipReadings)


MaxTemp = schemayearTemp_reading.groupBy('station')\
    .agg(F.max('value').alias('maxTemp'))

MaxTemp = MaxTemp.where((MaxTemp['maxTemp'] >=25) & (MaxTemp['maxTemp'] <=30))

MaxPrecip = schemaPrecipReadings.groupBy('station', 'date')\
    .agg(F.sum('value').alias('totDailyPrecipitation'))\
    .groupBy('station').agg(F.max('totDailyPrecipitation').alias('maxDailyPrecipitation'))

MaxPrecip = MaxPrecip.where( (MaxPrecip['maxDailyPrecipitation'] >= 100) &
(MaxPrecip['maxDailyPrecipitation'] <= 200))

stationList = MaxTemp.join(MaxPrecip, on = ['station'])\
    .select('station', 'maxTemp', 'maxDailyPrecipitation')\
    .orderBy('station', ascending=[False])

# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
stationList.rdd.saveAsTextFile("BDA/output")
```

Result

NULL

Question 5

year, month, avgMonthlyPrecipitation ORDER BY year DESC, month DESC

```
precipitation_file = sc.textFile("BDA/input/precipitation-readings.csv")
lines = precipitation_file.map(lambda line: line.split(";"))

precipReadings = lines.map(lambda p: Row(station=p[0],
                     date=p[1],
                     year=p[1].split("-")[0],
                     month=p[1].split("-")[1],
                     time=p[2],
                     value=float(p[3]),
                     quality=p[4]))

schemaPrecipReadings = sqlContext.createDataFrame(precipReadings)

# -----
```

```
station_file = sc.textFile("BDA/input/stations-Ostergotland.csv")
lines = station_file.map(lambda line: line.split(";"))

StationReadings = lines.map(lambda p: Row(station=p[0]))

schemaStationReadings = sqlContext.createDataFrame(StationReadings)

# ----
# schemaPrecipReadings.show()
# schemaStationReadings.show()

schemaPrecipReadings = schemaPrecipReadings.join(schemaStationReadings, on = ['station'])

schemaPrecipReadings = schemaPrecipReadings.where((schemaPrecipReadings["year"] >= 1993) &
(schemaPrecipReadings["year"] <= 2016))\
    .groupBy("year", 'month', 'station').agg(F.sum('value').alias('totMonPrecip'))\
    .orderBy(["year", 'month'], ascending=[False, False])\
    .groupBy("year", 'month').agg(F.avg('totMonPrecip').alias('avgMonthlyPrecipitation'))


# Following code will save the result into /user/ACCOUNT_NAME/BDA/output folder
schemaPrecipReadings.rdd.saveAsTextFile("BDA/output")
```

Result

```
Row(year='2016', month='07', avgMonthlyPrecipitation=0.0)
Row(year='2016', month='06', avgMonthlyPrecipitation=47.6625)
Row(year='2016', month='05', avgMonthlyPrecipitation=29.250000000000004)
Row(year='2016', month='04', avgMonthlyPrecipitation=26.900000000000006)
Row(year='2016', month='03', avgMonthlyPrecipitation=19.962500000000002)
Row(year='2016', month='02', avgMonthlyPrecipitation=21.5625)
Row(year='2016', month='01', avgMonthlyPrecipitation=22.325)
```